IN THE CLAIMS

Please amend the claims as indicated:

1. (currently amended) A method for detecting memory leaks in a software program, said method comprising the steps of:

monitoring a specified one or more analysis properties of software objects executing in the software program, wherein the one or more specified analysis properties include an object's age;

determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property, wherein a predetermined limit for an object's age is an object age limit;

identifying at least one software object determined to have one or more analysis properties that exceed that property's predetermined limit;

generating a stack walkback for the at least one identified software object; and

generating a statistics report including the generated stack walkback for the at least one identified software object, wherein said statistics report includes a notation for one or more software object classes that have instance counts that have grown by a factor of ten or more since a time period from when said software object classes are first reported within said statistics report, and wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that indicates a location of an executing logic at a time of the out-of-memory error, and wherein the generated statistics report identifies the likely location of at least one memory leak in the software program.

2. (previously presented) The method according to claim 1, further comprising calculating the object's age by timing a current period starting when the object was instantiated.

3. (previously presented) The method according to claim 1, wherein the one or more specified analysis properties include an object's instance count having a predetermined limit that is an object instance count growth value; and further comprising calculating the object instance count growth as a magnitude of growth of the object's instance count over a given time period.

4. (previously presented) The method according to claim 1, wherein monitoring comprises monitoring objects within a class designated for monitoring.

5. (canceled)

6. (canceled)

7. (canceled)

8. (previously presented) The method according to claim 1, further comprising generating a web interface for user viewing of the statistics report.

9. (original) The method according to claim 1, wherein the software objects are Java objects.

10. (previously presented) The method according to claim 1, further comprising:

monitoring an amount of available memory for a software program referencing the software objects;

determining when the amount of available memory for the software program referencing the software objects is within a predetermined threshold amount of memory from zero memory available for the software program utilizing the software objects; and

upon determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software objects, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for the software program referencing software objects dropping below an amount of available memory necessary to store the current stack walkback.

11. (currently amended) A system for detecting memory leaks in a software program comprising:

a processor; and

code executed by the processor including:

program means for monitoring a specified one or more analysis properties of software objects executing in the software program, wherein the one or more specified analysis properties include an object's age;

program means for determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property, wherein a predetermined limit for an object's age is an object age limit;

program means for identifying at least one software object determined to have one or more analysis properties that exceed that property's predetermined limit;

program means for generating a stack walkback for the at least one identified software object; and

program means for generating a statistics report including the generated stack walkback for the at least one identified software object, wherein said statistics report includes a notation for one or more software object classes that have instance counts that have grown by a factor of ten or more since a time period from when said software object classes are first reported within said statistics report, and wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that indicates a location of an executing logic at a time of the out-of-memory error, and wherein the generated statistics report and the generated stack walkback together identify the likely location of at least one memory leak in the software program.

12. (previously presented) The system according to claim 11, further comprising means for calculating the object's age by timing a current period starting when the object was instantiated.

13. (previously presented) The system according to claim 11, wherein the one or more specified analysis properties include an object's instance count having a predetermined limit that is an object instance count growth value; and further comprising means for calculating the object instance count growth as a magnitude of growth of the object's instance count over a given time period.

14. (original) The system according to claim 11, wherein the means for monitoring comprises monitoring objects within a class designated for monitoring.

15. (canceled)

16. (canceled)

17. (canceled)

18. (previously presented) The system according to claim 11, further comprising means for generating a web interface for user viewing of the statistics report.

19. (original) The system according to claim 11, wherein the software objects are Java objects.

20. (previously presented) The system according to claim 11, further comprising:

means for monitoring an amount of available memory for a software program referencing the software objects;

means for determining when the amount of available memory for the software program referencing the software objects is within a predetermined threshold amount of memory from zero memory available for the software program utilizing the software objects; and

means for, upon determining that the amount of available memory for the software program referencing the software objects is within the predetermined threshold amount of memory from zero memory available for the software program utilizing the software objects, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for the software program referencing software objects dropping below an amount of available memory necessary to store the current stack walkback.

21. (currently amended) An article of manufacture comprising a machine-readable storage device including program logic embedded therein for detecting memory leaks in a software program that causes control circuitry in a data processing system to perform the steps of:

monitoring a specified one or more analysis properties of software objects executing in the software program, wherein the one or more specified analysis properties include an object's age;

determining if any analysis property of software objects being referenced following a garbage collection process exceeds a respective predetermined limit for such analysis property, wherein a predetermined limit for an object's age is an object age limit;

identifying at least one software object determined to have one or more analysis properties that exceed that property's predetermined limit;

generating a stack walkback for the at least one identified software object; and

generating a statistics report including the generated stack walkback for the at least one identified software object, wherein said statistics report includes a notation for one or more software object classes that have instance counts that have grown by a factor of ten or more since a time period from when said software object classes are first reported within said statistics report, and wherein the statistics report is generated before the occurrence of an out-of-memory error and in a format that indicates a location of an executing logic at a time of the out-of-memory error, and wherein the generated statistics report identifies the likely location of at least one memory leak in the software program.

22. (previously presented) The article of manufacture of claim 21, further comprising the step of calculating the object's age by timing a current period starting when the object was instantiated.

23. (previously presented) The article of manufacture of claim 21, wherein the one or more specified analysis properties include an object's instance count having a predetermined limit that

is an object instance count growth value; and further comprising calculating the object instance count growth as a magnitude of growth of the object's instance count over a given time period.

24. (original) The article of manufacture of claim 21, wherein the step of monitoring comprises monitoring objects within a class designated for monitoring.

25. (canceled)

26. (canceled)

27. (canceled)

28. (previously presented) The article of manufacture of claim 21, further comprising the step of generating a web interface for user viewing of the statistics report.

29. (original) The article of manufacture of claim 21, wherein the software objects are Java objects.

30. (previously presented) The article of manufacture of claim 21, further comprising the steps of:

monitoring an amount of available memory for a software program referencing the software objects;

determining when the amount of available memory for the software program referencing the software objects is within a predetermined threshold amount of memory from zero memory available for the software program utilizing the software objects; and

upon determining that the amount of available memory for the software program referencing the software objects is within a predetermined threshold amount of memory from zero memory available for the software program utilizing the software objects, storing a current stack walkback of currently referenced software objects prior to the amount of available memory for the software program referencing software objects dropping below an amount of available memory necessary to store the current stack walkback.